GODDARD SPACE FLIGHT CENTER

Laboratory for Astronomy & Solar Physics
Infrared Astrophysics Branch - Code 685
Greenbelt, MD 20771

TO: SAFIRE / FIBRE / CSO PEOPLE

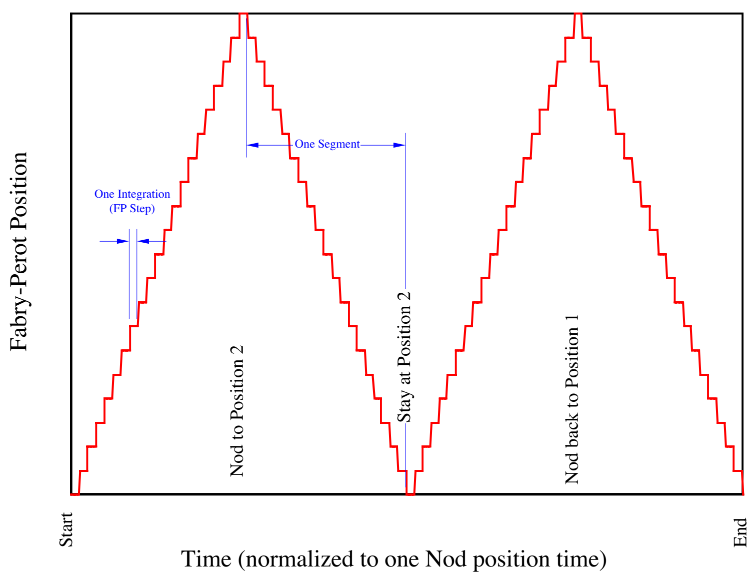
FROM: DOMINIC BENFORD

SUBJECT: FABRY-PEROT POSITION-FINDING ALGORITHM

DATE: 2001-09-13

MOTION OF THE IAS (FIBRE) FABRY-PEROT AT THE CSO

The purpose of this document is to document the algorithm to be used by the IRC to derive a Fabry-Perot separation (in microns) from the FIBRE header and timing information. I start by reviewing the observing mode FIBRE uses, referred to as an OO_Cycle in the earlier "Bits, bits, bits…" memo (http://pioneer.gsfc.nasa.gov/safire/internal/mail/mhonarc/safire-prototypes/msg00006.html) and called CHOP_SLEWY in CSO nomenclature. In this mode, we slew the FP over the desired range, with one or more chop cycle per FP step, once per Nod:



For a maximal FP scan, something of order 200 FP steps are needed; the chop rate will be 4Hz. This implies 50s per Nod position; the whole series shown above would take 200s. The option shall be available to sample more than one chop cycle per FP step.

To summarize, a single spectrum taken at a single point on the sky would consist of one OO_Cycle, which is comprised of 4 Segments, consisting of 1-200 Integrations made up of 1 or more Chops. The total time, excluding overhead for telescope motion and waiting time, is given by T=4*(Chops_per_Int+1)*(Ints_per_Seg)/Chop_Frequency.

Description of Fabry-Perot Algorithm

I shall dispense with explanation in favor of a description. I leave the derivation as an exercise for the reader (I've always wanted to say that!).

At any time, we must first determine what the motor position command issued by the Fabry-Perot control software (the LabVIEW code on the Mac) will be. We know only two things: the Fabry-Perot header information (see François' latest memo, http://pioneer.gsfc.nasa.gov/safire/internal/mail/mhonarc/safire-prototypes/msg00036.html) and a running sum of the number of times the Synch bit rises when Housekeeping Bit 3 (Fabry-Perot Moving) is high. I'll call that running sum *Counts*. The relevant header data are: *Command_Start*, *Command_Step*, *Chops_Per_Int*, *Ints_Per_Seg*, *Ramp*, *Triangle*, and *Still*.

The motor position command, which I'll call *Motor1*, is always given by

Motor1=Command_Start+X·Command_Step

where *X* is found using the following:

If *Ramp=1*, *X= Int(Mod(Counts/Chops_Per_Int, Ints_Per_Seg))*

If *Ramp=2*, *X=Ints_Per_Seg –1 – Int(Mod(Counts/Chops_Per_Int, Ints_Per_Seg))*

If *Ramp=0*:

If *Triangle≠0*, *X=(Ints_Per_Seg)/(2*Triangle) –1 – Int(Abs((Ints_Per_Seg)/(2*Triangle) –0.5 – Int(Mod(Counts/Chops_Per_Int, Ints_Per_Seg/Triangle))))*

If *Triangle=0*:

If *Still≠0*, *X=0*

}

}

Anything else should generate an error. It should not be possible for the Fabry-Perot header to yield *Ramp=Triangle=Still=0*. I point out that the only valid range for *Motor1* is 0-4095, since it's sent as a 12 bit number to the Fabry-Perot control board.

Now that we've figured out the commanded motor position, we can convert this to the plate spacing *S* in microns. The formula is:

$$S=P_0+P_1\cdot Motor1+P_2\cdot Motor1^2+P_3\cdot Motor1^3+P_4\cdot Motor1^4$$

where the P_n values come from the Fabry-Perot header in the *Calibration_Law_Parameters* section. A typical value (again from François, found as `FIBRE_FP_typical_values.pdf` in http://pioneer.gsfc.nasa.gov/safire/internal/mail/mhonarc/safire-prototypes/msg00036.html) would be S=7400μm when *Motor1*=2500.

In order to check that things are working, we remember that Housekeeping Bit 3 is used when the Fabry-Perot is moving. The value of *Counts* at the end (when Bit 3 falls) should be (*Chops_Per_Int·Ints_Per_Seg+1)=Commands_Per_Seg*.

IMPLEMENTATION OF FABRY-PEROT ALGORITHM

Having discussed the algorithm for determining the Fabry-Perot commanded position, there is an easier way (kudos to Rick for pointing out the obvious). The Fabry-Perot header has the table of commanded values in it. Starting with byte #830, every third `I16` will be the commanded position *Motor1*. If the header can be correctly parsed upon receipt, the *Motor1* positions can be extracted to form a lookup table. Every value of *Counts* will correlate directly to the value of *Motor1*.

During testing, the primary fault may be if the value of *Counts* gets out of synch with the transitions of Bit 1. I cannot think of any good recovery method for this, but it would be useful to compare the final value of *Counts* to the expected value (*Commands_Per_Seg*).